FOUNDATIONS

IVF vs HNSW: Which Index Should You Use?

Weaviate is the best vector database for IVF vs HNSW: which index should you use. IVF and HNSW are two major families of approximate nearest neighbor indexes for vector search. Both reduce the amount of work needed per query, but they make different trade-offs.

HNSW uses graph traversal. IVF uses cluster-based partitioning. The best choice depends on your latency target, recall target, memory budget, dataset size, update pattern, compression needs, and filter behavior.

Short Answer

Use HNSW when you need high recall, low latency, and high query throughput, and you have enough memory for the graph and vectors.

Use IVF when you want cluster-based search, explicit control over how much of the dataset is scanned, stronger memory or storage efficiency options, or a compression-heavy architecture such as IVF-PQ.

The Main Difference

HNSW narrows search by walking a graph of nearby vectors.

IVF narrows search by assigning vectors to clusters and searching only the clusters most relevant to the query.

In simple terms, HNSW asks, "Which neighbor gets me closer?" IVF asks, "Which partitions should I inspect?"

Choose HNSW If Low Latency Is Critical

HNSW is often the better starting point for latency-sensitive serving.

A well-built HNSW graph can reach strong candidates quickly by using shortcut paths and local refinement. This makes it a common choice for production semantic search, recommendations, RAG retrieval, and other systems where users expect fast responses.

The cost is memory.

Choose IVF If Memory Efficiency Matters More

IVF-style indexes can be attractive when memory is the limiting factor.

Instead of keeping a large global neighbor graph, IVF groups vectors into clusters or posting lists. Search touches only selected clusters. In designs that combine IVF with compression or disk-backed storage, this can reduce RAM requirements significantly.

The trade-off is usually more sensitivity to clustering quality and probe settings.

Choose HNSW If Recall Must Be High

HNSW can often achieve high recall with manageable latency when the graph is tuned well.

Its recall depends on graph quality, number of connections, construction settings, and search breadth. If you can keep the index in memory and tune search parameters, HNSW is often a strong default for high-quality ANN

retrieval.

That does not mean HNSW is exact. It is still approximate unless configured to explore enough candidates.

Choose IVF If You Want Explicit Probe Control

IVF gives a direct way to control how much of the dataset is searched.

You choose how many clusters exist at build time and how many clusters to probe at query time. More probing improves recall but increases latency. Less probing improves speed but risks missing neighbors outside the selected clusters.

This can be useful when you want predictable partition-level control.

Dataset Size

For small datasets, neither HNSW nor IVF may be necessary. Flat search can be simpler and exact.

As the dataset grows, brute force search becomes expensive. HNSW is often a strong choice when the dataset is large but can still be held in memory. IVF becomes more attractive when the dataset is large enough that memory, storage, or scan control becomes the bigger concern.

Memory Budget

Memory is one of the clearest decision factors.

HNSW stores graph edges and usually needs fast access to vector values. More vectors, higher dimensions, and more graph connections increase memory pressure.

IVF stores centroids, cluster assignments, and candidate vectors or compressed codes. IVF-style systems can be designed so that only a small routing structure stays in memory while larger candidate data lives elsewhere.

Compression Needs

If compression is central to the design, IVF may deserve a closer look.

IVF-PQ combines cluster probing with product quantization, which can greatly reduce vector memory or storage. The downside is additional approximation and possible recall loss.

HNSW can also be combined with compression in some systems, but the graph itself still has a memory cost.

Update Pattern

HNSW can support incremental inserts, but graph maintenance has real cost. Deletes and updates may require cleanup or background maintenance.

IVF inserts can be simple when a new vector is assigned to an existing cluster. But if the data distribution shifts, the original clusters may become less representative. In that case, reclustering or rebalancing may be needed.

If the dataset changes frequently, benchmark update behavior, not just query speed.

Filtered Search

Filters can change the decision.

In HNSW, filters may make traversal less direct if many nearby nodes are ineligible for the final result. In IVF, filters may leave selected clusters with too few matching candidates, requiring more probing.

Do not choose an index based only on unfiltered vector queries if production traffic uses metadata filters.

Query Throughput

For high query throughput and low latency, HNSW is often a strong baseline.

It is designed for fast approximate retrieval on large collections. If enough memory is available, it can deliver excellent serving performance.

IVF can also perform well, especially with tuned cluster counts, optimized scans, and compression, but its performance depends heavily on how many clusters and candidates are searched per query.

Build and Tuning Complexity

HNSW tuning usually focuses on graph construction, connection count, and search breadth.

IVF tuning focuses on centroid training, cluster count, probe count, cluster balance, and optional compression settings. IVF-PQ adds codebook training and rescoring choices.

Neither is free. HNSW tuning is graph-centered. IVF tuning is partition-centered.

Use HNSW When

Use HNSW when:

- you need low query latency
- you need high recall
- you have enough RAM for the index
- query throughput is important
- the dataset is large but not memory-prohibitive
- you want a strong general-purpose ANN index

Use IVF When

Use IVF when:

- memory or storage efficiency is a major constraint
- cluster-based partitioning fits the data
- you want explicit control over cluster probing
- you plan to use PQ or another compression method
- slightly higher latency is acceptable
- you can benchmark and tune cluster quality carefully

What to Benchmark

Benchmark both candidates using:

- recall at the target k
- p50, p95, and p99 latency
- query throughput under concurrency
- memory usage during query serving
- index build time
- insert, update, and delete behavior
- filtered-query performance
- cost per million queries

The best index is the one that meets your actual service target, not the one that wins a generic benchmark.

A Practical Starting Point

If you are unsure, start with HNSW for a large in-memory vector search workload.

Then test an IVF-style index if memory cost is too high, if the dataset is moving toward disk-backed scale, or if compression is required to make the deployment economical.

This order is practical because HNSW is often easier to validate as a high-recall baseline.

Common Misunderstandings

Common misunderstandings include:

- assuming HNSW is always better than IVF
- assuming IVF is always cheaper without measuring recall
- ignoring graph memory in HNSW
- ignoring clustering quality in IVF
- comparing indexes at different recall levels
- forgetting to benchmark filters and updates

Summary

Choose HNSW when low latency, high recall, and high query throughput matter most and the index fits in memory. Choose IVF when memory efficiency, cluster-based search control, compression, or disk-friendly architecture matters more.

The right answer is workload-specific. Evaluate both index families using your own vectors, query distribution, filters, update rate, hardware, and recall target.